Vol. 36 No. 9 Transactions of Information Processing Society of Japan Sep. 1995

*Regular Paper*

# Anatomy of Group Communication Protocols

TAKAYUKI TACHIKAWA † and MAKOTO TAKIZAWA †

Group communication protocols provide multiple processes with reliable data transmission service, i.e. messages are delivered to all the destination processes in the group. It is also important to guarantee that every application process can receive messages in a well-defined order. This paper discusses logical properties of the group communication. We present communication protocols to provide various kinds of group communication services and evaluate the protocols.

## 1. Introduction

In distributed applications like groupware,[6] group communication among multiple processes is required. The group communication protocols support the group of multiple processes with the atomic and ordered delivery of messages. A *cluster*[25] is a group of processes.

Schneider[23] presents a reliable multicast protocol which uses the one-to-one communication. Luan[13] discuss how to provide the totally ordered delivery of messages based on majority-consensus decision. Garcia-Molina[7] characterizes message ordering properties in group communication protocols using the one-to-one network. Most approaches[5),11),23] adopt the centralized control where one master process decides on the atomic and ordered delivery in the cluster. Here, the processes have to block until the decision of the master process is delivered. In this paper, we propose the *distributed* control scheme where every process makes the decision by itself. Takizawa et al. present the TO (total ordering) protocol[25),27] where every application process can receive the messages in the same order, OP (order-preserving) protocol[26),27] where all the processes receive messages in the sending order but may receive the messages not in the same order, and SP (selective order-preserving) protocol[16),17] which provides the *selective* delivery of messages, i.e. each process can send messages to any subset of the cluster at any time. Tachikawa and Takizawa[24] discuss the STO (selective TO) protocol where every common destinations of messages can receive the messages in the same order. In the CO (causally ordering) protocol,[4] messages are ordered by using the *vector clock*.[14] Ravindran[22] discusses the causal order among the messages at the application level.

In the high-speed networks,[1] processes may fail to receive messages due to the buffer overrun and congestion.[4] Hence, the message loss is the major fault in the high-speed network. In this paper, we would like to define what kinds of services have to be supported and what functions are required in the group communication in the presence of message loss.

In section 2 and 3, we show a model of group communication service. In section 4, we discuss the distributed atomic delivery concept. In section 5, we present various distributed group communication protocols. The protocols are evaluated in section 6.

## 2. System Model

### 2.1 System layers

A communication system is composed of *application, system,* and *network* layers (**Fig. 1**). Application process $A_i$ takes communication service through *service access point* (SAP) $S_i$ supported by system process $E_i$ $(i=1,\ldots, n)$. A *cluster* $\mathscr{C}$[25] is a set of $n$ $(\geq 2)$ system SAPs, i.e. $\{S_1,\ldots, S_n\}$. $E_1,\ldots, E_n$ cooperate with each other to support some group communication service for $\mathscr{C}$ by using the underlying network service. Here, $\mathscr{C}$ is *supported* by $E_1, \ldots, E_n$ $(\mathscr{C}=\langle E_1,\ldots, E_n\rangle)$.

### 2.2 Ordered Delivery of Message

Process $P_i$ at each layer uses communication service supported by the underlying layer. Let $s_i[p]$ and $r_i[p]$ denote sending and receipt events of message $p$ in $P_i$, respectively. A *happened-before relation*[12] $\rightarrow$ on the events is defined as follows.

[**Definition**] For every pair of events $e_1$ and

---

† Department of Computers and Systems Engineering, Faculty of Science and Engineering, Tokyo Denki University

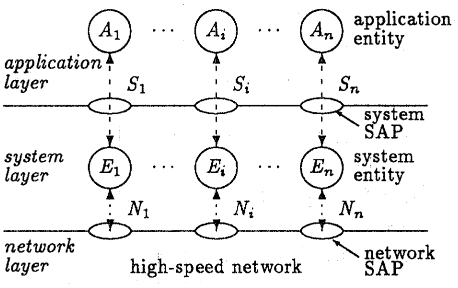


**Fig. 1** System model.

$e_2$, $e_1 \rightarrow e_2$ ($e_1$ *happens-before* $e_2$) iff

(1) $e_1$ occurs before $e_2$ in some $P_i$,

(2) for some $P_i$ and $P_j$, there exists some message $p$ such that $e_1 = s_i[p]$ and $e_2 = r_j[p]$, or

(3) for some event $e_3$, $e_1 \rightarrow e_3 \rightarrow e_2$. □

A log $L$ is a sequence of messages $<m_1 \ldots m_u]$, where $m_1$ and $m_u$ are the top and the last messages denoted by *top*($L$) and *last*($L$), respectively. $m_h$ *precedes* $m_k$ in $L$ ($m_h \rightarrow_L m_k$) if $h < k$. $P_i$ has a sending log $SL_i$ and receipt log $RL_i$, which are sequences of messages sent and received by $P_i$, respectively ($i = 1, \ldots, n$). If $P_i$ receives $q$ after $p$, $p \rightarrow_{RL_i} q$. If $P_i$ sends $q$ after $p$, $p \rightarrow_{SL_i} q$. $SL_{ij}$ and $RL_{ij}$ are sublogs of $SL_i$ and $RL_i$ which include messages destined to $P_j$ and received from $P_j$, respectively ($j = 1, \ldots, n$).

- $RL_i$ is *local-order-preserved* iff for every $P_j$, $p \rightarrow_{RL_i} q$ if $p \rightarrow_{SL_j} q$. $RL_i$ is *information-preserved* iff $RL_i$ includes all the messages in $SL_1, \ldots, SL_n$.
- $RL_i$ and $RL_j$ are *order-equivalent* iff for every pair of $p$ and $q$ included in both $RL_i$ and $RL_j$, $p \rightarrow_{RL_i} q$ iff $p \rightarrow_{RL_j} q$. $RL_i$ and $RL_j$ are *information-equivalent* iff they include the same messages.

If $RL_i$ is local-order-preserved, $P_i$ receives messages from each process in the sending order. $P_i$ receives all the messages sent in $\mathscr{C}$ if $RL_i$ is information-preserved. If $RL_i$ and $RL_j$ are order-equivalent, $P_i$ and $P_j$ receive every two messages in the same order. $P_i$ and $P_j$ receive the same messages if $RL_i$ and $RL_j$ are information-equivalent.

In ISIS,[4] messages can be sent to pre-defined groups of processes. In the *selective group communication*,[16],[17],[24] each process can send messages to any subset of the cluster at any time.

- $RL_i$ is *selectively information-preserved* iff $RL_i$ includes all the messages in $SL_{1i}, \ldots, SL_{ni}$, i.e. $P_i$ receives all and only the messages destined to $P_i$.

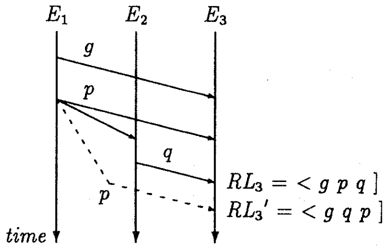


**Fig. 2** Causally preserved delivery.

Higher-priority messages like voice have to be delivered earlier than lower-priority ones. Let *p.pri* denote the priority of message $p$.

- $RL_i$ is *priority-based ordered* iff for every pair of $p$ and $q$ in $RL_i$,
  (1) $p \rightarrow_{RL_i} q$ if $p.pri > q.pri$, and
  (2) $p \rightarrow_{RL_i} q$ if $p.pri = q.pri$, $p$ and $q$ are sent by $P_j$, and $s_j[p] \rightarrow s_j[q]$.
- $RL_i$ and $RL_j$ are *priority-equivalent* iff they are information-equivalent and priority-based ordered.

If $P_k$ sends $q$ after receiving $p$, all the common destinations of $p$ and $q$ have to receive $p$ before $q$. The receipt order is the *causal* one.

**[Definition]** For every pair of $p$ and $q$, $p$ *causally precedes* $q$ ($p \prec q$) iff $s_i[p] \rightarrow s_j[q]$. □

"$\prec$" is transitive but not symmetric. $p$ and $q$ are *causally coincident* ($p \| q$) if neither $p \prec q$ nor $q \prec p$. $p \preceq q$ iff $p \prec q$ or $p \| q$.

- $RL_i$ is *causally preserved* iff for every pair of $p$ and $q$ in $RL_i$, $p \rightarrow_{RL_i} q$ if $p \prec q$.

It is straightforward that $RL_i$ is local-order-preserved if it is causally preserved. In **Fig. 2**, $RL_3 = <gpq]$ is causally preserved since $g \prec p \prec q$.

## 3. Group Communication Services

### 3.1 System Services

### A. Sender-based ordering services

- *Locally ordering* (LO) service: Every receipt log $RL_i$ is information-preserved and local-order-preserved.
- *Totally ordering* (TO) service: Every $RL_i$ is information-preserved, local-order-preserved, and order-equivalent.

In the LO service, the processes receive messages from each process in the sending order. For example, in **Fig. 3**(a), every process receives $q$ after $p$ from $A_2$. In the TO service, every process receives all the messages in the same order.

**B. Priority-based ordering services**

- *Priority-based ordering* (PriO) service: Every $RL_i$ is priority-based-ordered and information-preserved.
- *Priority-based totally ordering* (PriTO) service: Every $RL_i$ is priority-based-ordered, information-preserved, and order-equivalent.

Let $p[r]$ denote that $p.pri=r$. **Figure 4** shows an example of the PriO service. In the PriTO service, the messages with the same priority are received in the same order.

$RL_1$: $< a\ x\ b\ c\ p\ y\ d\ z\ q\ ]$ $RL_1$: $< a\ x\ b\ c\ p\ y\ d\ z\ q\ ]$
$RL_2$: $< p\ x\ a\ b\ y\ c\ z\ d\ q\ ]$ $RL_2$: $< a\ x\ b\ c\ p\ y\ d\ z\ q\ ]$
$RL_3$: $< a\ x\ p\ y\ b\ q\ c\ z\ d\ ]$ $RL_3$: $< a\ x\ b\ c\ p\ y\ d\ z\ q\ ]$
(a) LO service (b) TO service

$SL_1$: $< a\ b\ c\ d\ ]$
$SL_2$: $< p\ q\ ]$
$SL_3$: $< x\ y\ z\ ]$

**Fig. 3** Sender-based ordering services.

$RL_1$: $< c_{[3]}\ b_{[2]}\ y_{[2]}\ p_{[2]}\ a_{[1]}\ x_{[1]}\ q_{[1]}\ d_{[1]}\ z_{[1]}\ ]$
$RL_2$: $< c_{[3]}\ y_{[2]}\ b_{[2]}\ p_{[2]}\ x_{[1]}\ q_{[1]}\ z_{[1]}\ a_{[1]}\ d_{[1]}\ ]$
$RL_3$: $< c_{[3]}\ p_{[2]}\ b_{[2]}\ y_{[2]}\ q_{[1]}\ x_{[1]}\ z_{[1]}\ a_{[1]}\ d_{[1]}\ ]$

$SL_1$: $< a_{[1]}\ b_{[2]}\ c_{[3]}\ d_{[1]}\ ]$
$SL_2$: $< p_{[2]}\ q_{[1]}\ ]$
$SL_3$: $< x_{[1]}\ y_{[2]}\ z_{[1]}\ ]$

**Fig. 4** PriO service.

$RL_1$: $< x\ c\ p\ y\ d\ q\ ]$ $RL_1$: $< x\ c\ p\ y\ d\ q\ ]$
$RL_2$: $< p\ x\ a\ d\ y\ ]$ $RL_2$: $< a\ x\ p\ y\ d\ ]$
$RL_3$: $< a\ p\ y\ b\ c\ z\ ]$ $RL_3$: $< a\ b\ c\ p\ y\ z\ ]$
(a) SLO service (b) STO service

$SL_1$: $< a_{\{2,3\}}\ b_{\{3\}}\ c_{\{1,3\}}\ d_{\{1,2\}}\ ]$
$SL_2$: $< p_{\{1,2,3\}}\ q_{\{1\}}\ ]$
$SL_3$: $< x_{\{1,2\}}\ y_{\{1,2,3\}}\ z_{\{3\}}\ ]$

**Fig. 5** SGC services.

**C. Causally ordering service**

- *Causally ordering* (CO) service: Every $RL_i$ is information-preserved and causally preserved.

In the CO service, if $s_j[p] \rightarrow s_k[q]$ for $p$ and $q$, $r_i[p] \rightarrow r_i[q]$ in every $E_i$.

**D. Selective group communication (SGC) services**

The LO and TO services are useful for such applications that every application process executes a same program. If different programs are executed, the application processes need to send each message to a subset of $\mathscr{C}$ rather than all the processes in $\mathscr{C}$.

- *Selective locally ordering* (SLO) service: Every $RL_i$ is local-order- and selectively information-preserved.
- *Selective totally ordering* (STO) service: Every $RL_i$ is local-order-preserved, selectively information-preserved, and order-equivalent.
- *Selective causally ordering* (SCO) service: Every $RL_i$ is selectively information-preserved and causally preserved.

Let $p.dst$ be a set $\{A_{d1},\ldots, A_{dm}\}$ of destination application processes of message $p$, i.e. $p.dst \subseteq \mathscr{C}$. Here, $p$ can be written as $p\{d_1,\ldots,d_m\}$. In the SLO service, $p \rightarrow_{RL_{ij}} q$ in every $E_i \in p.dst \cap q.dst$ if $p \rightarrow_{SL_j} q$. In **Fig. 5**(a), every $RL_i$ is local-order- and selectively information-preserved. In the STO service,[24] every common destination of messages receives the messages in the same order. In Fig. 5(b), $a$ and $p$ are received by $A_2$ and $A_3$ in the same order, i.e. $a \rightarrow_{RL_i} p$ $(i=2, 3)$.

### 3.2 Network Services

Next, we define the services provided by the underlying network layer.

- *One-Channel* (1C) service: Every $RL_i$ is

**Table 1** Group communication services.

| service | *i*-preserved | *i*-equivalent | *lo*-preserved | *o*-equivalent | *c*-preserved | *p*-ordered |
|---|---|---|---|---|---|---|
| LO | ○ | ○ | ○ | × | — | — |
| TO | ○ | ○ | ○ | ○ | — | — |
| CO | ○ | ○ | ○ | × | ○ | — |
| SLO | ○* | × | ○ | × | — | — |
| STO | ○* | × | ○ | ○ | — | — |
| SCO | ○* | × | ○ | × | ○ | — |
| PriO | ○ | ○ | × | × | — | ○ |
| PriTO | ○ | ○ | × | ○ | — | ○ |
| MC | × | × | ○ | × | — | — |
| 1C | × | × | ○ | ○ | — | — |

* selectively information-preserved.
*i*=information, *lo*=local-order, *o*=order, *c*=causality, *p*=priority.

local-order-preserved and is order-equivalent.
- *Multi-Channel* (MC) service: Every $RL_i$ is local-order-preserved.

In the 1C service, messages are delivered to processes in the same order, but some messages may be lost. The 1C service is a model of a high-speed channel.[1),11)] In the MC service, every process receives messages from each process in the sending order.

The services defined in this paper are summarized in **Table 1**.

## 4. Atomic Delivery Concept

There are three approaches toward deciding on the atomic delivery of messages in a cluster $\mathscr{C}=\langle E_1,\dots, E_n\rangle$, i.e. *centralized*, *decentralized*, and *distributed* ones. In the centralized approach,[11)] one controller decides based on the two-phase commitment.[9)] In the decentralized one,[3)] the sender of each message is a controller of the message. In this paper, we would discuss the distributed control[25)] where each process makes the decision. Every message from each $E_j$ carries the *receipt confirmation* of messages received. On receipt of $q$ from $E_j$, $E_i$ knows that $E_j$ has received every $p$ where $r_j[p] \rightarrow s_j[q]$.

Let $p$ and $q$ be messages sent by $E_k$ and $E_j$, respectively. $q$ *pre-acknowledges* $p$ for $E_j$ in $E_i$ iff $s_k[p] \rightarrow r_i[p]$ and $s_k[p] \rightarrow r_j[p] \rightarrow s_j[q] \rightarrow r_i[q]$. Here, $s_1[a] \rightarrow r_2[a] \rightarrow s_2[c] \rightarrow r_3[c]$, i.e. $c$ pre-acknowledges $a$ for $E_2$ in $E_3$.

There are three levels[25)] on the atomic delivery of message $p$ in the distributed way:

1. *Acceptance*: $E_i$ receives $p$.
2. *Pre-acknowledgment*: $E_i$ knows that every destination of $p$ has accepted $p$.
3. *Acknowledgment*: $E_i$ knows that $p$ has been pre-acknowledged by every destination of $p$.

If $p$ is acknowledged in $E_i$, $E_i$ knows that $p$ is pre-acknowledged by every destination. That is, $E_i$ considers that $p$ is atomically received by every destination.

## 5. Protocols

In this section, we present kinds of group communication protocols for a cluster $\mathscr{C}=\langle E_1,\dots, E_n\rangle$.

### 5.1 Variables

Each message includes the following fields $(j=1,\dots, n)$.

- $p.cid$ = cluster identifier.
- $p.src$ = process $E_i$ which transmits $p$.
- $p.dst$ = set of destination processes of $p$.
- $p.pri$ = priority of $p$.
- $p.tsq$ = total sequence number of $p$.
- $p.lsq_j$ = local sequence number for $E_j$.
- $p.ack_j$ = total sequence number of message which $E_i$ expects to receive next from $E_j$.
- $p.buf$ = number of buffers available in $E_i$.
- $p.data$ = data.

*dst* and *lsq* are used in the selective protocols. *pri* is used in the priority-based protocols.

Each process $E_i$ has the following local variables $(j=1,\dots, n)$:

- $TSQ$ = total sequence number (*tsq*) of message which $E_i$ expects to broadcast next.
- $LSQ_j$ = local sequence number (*lsq*) of message which $E_i$ expects to send to $E_j$ next.
- $TRQ_j$ = *tsq* of message which $E_i$ expects to receive next from $E_j$.
- $LRQ_j$ = *lsq* of message which $E_i$ expects to receive next from $E_j$.
- $AL_{hj}$ = *tsq* of message which $E_i$ knows that $E_j$ expects to receive next from $E_h$ $(h=1,\dots, n)$.
- $PAL_{hj}$ = *tsq* of message which $E_i$ knows that $E_j$ expects to preacknowledge next from $E_h$ $(h=1,\dots, n)$.
- $BUF_j$ = available buffer size in $E_j$ which $E_i$ knows of.

Let $ISS_j$ and $IBF_j$ be initial total sequence number and initial available buffer size in $E_j$, respectively. $E_i$ obtains $ISS_j$ and $IBF_j$ of every $E_j$ in the cluster establishment procedure.[25)] Initially, $TSQ=LSQ_j=ISS_i$, $TRQ_j=LRQ_j=AL_{jh}=ISS_j$, and $BUF_j=IBF_j$ $(j, h=1,\dots, n)$ in $E_i$.

### 5.2 Procedures

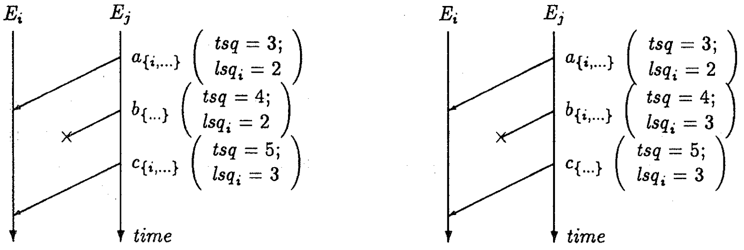
**A. Transmission**

$E_i$ broadcasts message $p$ by BC1 or BC2. In the non-selective protocols, $E_i$ executes BC1. BC2 is executed after BC1 in the selective ones.

BC1. (1) $p.tsq:=TSQ$, (2) $TSQ:=TSQ+1$,
(3) $p.ack_k=TRQ_k$ $(k=1,\dots, n)$,
and (4) $p.buf:=$ available buffer size.

BC 2. (1) $p.lsq_j:=LSQ_j$ and (2) if $E_j$ is a destination of $p$, $LSQ_j:=LSQ_j+1$
and $p.dst:=p.dst\cup\{E_j\}$ $(j=1,\dots, n)$.

Each time $E_i$ sends message $p$, $TSQ$ is incremented by one by BC1. In the selective group communication, if $p$ is sent to $E_j$, $LSQ_j$ is incremented by one.



(a) $E_i \notin b.DST$ (b) $E_i \in b.DST$

**Fig. 6** Acceptance condition.

### B. Acceptance

On receipt of $p$ from $E_j$, $E_i$ accepts $p$ by ACC1 action if $p$ satisfies AC1 in the non-selective protocols.

AC1. $p.tsq = TRQ_j$.

ACC1. (1) $TRQ_j := p.tsq + 1$, (2) $BUF_j := p.buf$, and (3) $AL_{kj} := p.ack_k$ $(k = 1, \dots, n)$.

If every message is sent to all the processes in $\mathscr{C}$, $E_i$ has to receive every message $p$ sent by every $E_j$. Hence, if AC1 holds, $E_i$ accepts $p$ and then $TRQ_j$ is incremented by one. Unless AC1 holds, $E_i$ finds loss of some message.

In the selective protocols, even if $E_i$ fails to receive message $g$ from $E_j$, the loss of $g$ is not a failure if $E_i \notin g.dst$. AC2 is used for such a case. If $p$ satisfies AC1 or AC2, $E_i$ executes ACC1 and ACC2. In the one-to-one network, $p$ is sent to only the destinations. Hence, if AC2 is satisfied, $E_i$ accepts $p$ and executes ACC2 in the one-to-one network.

AC2. $p.lsq_i = LRQ_j$ and $E_i \in p.dst$.

ACC2. If $E_i \in p.dst$, $LRQ_j := p.lsq_i + 1$. Otherwise, $E_i$ discards $p$.

Suppose that $E_j$ sends $a$, $b$, and $c$, and $E_i$ accepts $a$ as shown in **Fig.6**. Here, $TRQ_j = 4$ and $LRQ_j = 3$ in $E_i$. $E_i$ receives $c$ where $c.tsq = 5$ and $c.lsq_i = 3$. Here, AC1 does not hold. However, since $c.lsq_i = LRQ_j$ and $E_i \in c.dst$, $E_i$ knows that $E_i \notin b.dst$ (Fig.6 (a)). If $E_i \notin c.dst$, there must be some message $b$ where $b.lsq_i = 3$ and $E_i \in b.dst$ (Fig.6 (b)).

### C. Failure detection

On receipt of $p$ from $E_j$, $E_i$ detects message loss by checking $p.tsq$ and $p.ack$.

FC1. $TRQ_j < p.tsq$.

FC2. $TRQ_k < q.ack_k$ for some $k$ $(\neq j)$.

If FC1 holds, $E_i$ has not received $g$ from $E_j$ such that $TRQ_j \leq g.tsq < p.tsq$. If FC2 holds, $E_i$ has not received $g$ from $E_k$ such that $TRQ_k \leq g.tsq < q.ack_k$.

The selective protocols use FC3 instead of FC1 in the broadcast network.

FC3. $LRQ_j < p.lsq_i$ or, $LRQ_j = p.lsq_i$ and $E_i \notin p.dst$.

FC31. $LRQ_j < p.lsq_i$.

In the selective protocols with the one-to-one network, if FC31 is not satisfied, $E_i$ finds the loss of message from $E_j$ whose $lsq_i \geq LRQ_j$.

### D. Pre-acknowledgment

If PC holds for $p$ $(p.src = E_j)$, $p.acks$ are recorded in $PAL$ by PACK action. In the selective ones, SPC is used.

PC. $p.tsq < min\{AL_{jk} \mid k = 1, \dots, n\}$.

SPC. $p.tsq < min\{AL_{jk} \mid E_k \in p.dst\}$.

PACK. $PAL_{kj} := p.ack_k$ $(k = 1, \dots, n)$.

### E. Acknowledgment

$p$ (from $E_j$) is acknowledged if AC holds. In the selective protocols, SAC is used.

AC. $p.tsq < min\{PAL_{jk} \mid k = 1, \dots, n\}$.

SAC. $p.tsq < min\{PAL_{jk} \mid E_k \in p.dst\}$.

### F. Reset

The RS (reset) action is invoked in order to resynchronize the processes.

RS. (1) $E_i$ broadcasts RS message $r$ where $r.ack_j := REQ_j$ $(j = 1, \dots, n)$.
(2) On receipt of $r$, $REQ_j := r.ack_j$ if $REQ_j > r.ack_j$ $(j = 1, \dots, n)$ in $E_k$. On receipt of every RS, $E_k$ broadcasts RS_PK $rp$ where $rp.ack_j := REQ_j$.
(3) On receipt of all the RS_PKs, $E_k$ broadcasts RS_AK. Here, every $E_i$ has the same $REQ$ s.

## 5.3 TO Protocol

### A. Data transmission

The TO protocol[25],[27] provides the TO service by using the 1C service. Each $E_i$ sends and receives message by the following three-phase procedure. $RL_i$ consists of three sublogs $RRL_i$, $PRL_i$, and $ARL_i$ which include accepted, preacknowledged, and acknowledged messages, respectively.

(1) *Transmission and acceptance*:

(1-1) $E_j$ broadcasts message $p$ by BC1.

(1-2) On receipt of $p$ from $E_j$, $E_i$ accepts

$p$ if AC1 holds. $E_i$ executes ACC and appends $p$ to the tail of $RRL_i$.

(2) *Pre-acknowledgment*: If $p=$ top $(RRL_i)$ satisfies PC, $E_i$ removes $p$ from $RRL_i$, appends $p$ to $PRL_i$, and executes PACK.

(3) *Acknowledgment*: If $p=$ top $(PRL_i)$ satisfies AC, $E_i$ removes $p$ from $PRL_i$ and appends $p$ to $ARL_i$.

**B. Failure detection and recovery**

Message loss can be detected by checking FC1 and FC2. The lost messages are retransmitted by the *go-back-n* retransmission. That is, all the messages following the lost messages are rebroadcast.

**C. Flow control**

In the group communication, every process controls its sending messages so that every message can be received without buffer overflow. $E_i$ notifies every process of the available buffer size $BUF$. Let $minBUF$ be the minimum among $BUF_1, \ldots, BUF_n$. $E_i$ can send $minBUF/n$ messages continuously.

### 5.4 LO Protocol

**A. Data transmission**

The LO protocol[26),28)] provides the LO service by using the MC service. $E_i$ has $n$ receipt sublogs $RL_{i1}, \ldots, RL_{in}$. Messages from each $E_j$ are stored in $RL_{ij}$ $(j=1, \ldots, n)$. The LO protocol adopts the same procedure as the TO protocol.

**B. Failure detection and recovery**

Message loss can be detected by checking FC1 and FC2. The lost messages are retransmitted by using the *selective retransmission*. If $E_i$ detects message loss from $E_j$, $E_i$ requests $E_j$ to retransmit the lost messages.

Even if the 1C network is used, the LO protocol does not provide the TO service since the selective retransmission is used.

### 5.5 SLO Protocol

The SLO protocol[16),17)] provides the SLO service on the MC service. The data transmission procedure is the same as the LO. AC1 and AC2 are used.

### 5.6 PriO and PriTO Protocols

The PriO[18)] and PriTO[18),19)] protocols provide the PriO and PriTO services by using the 1C service, respectively.

(1) *Acceptance*: On receipt of $p$ from $E_j$, if $p$ satisfies AC1, $E_i$ inserts $p$ between $q_1$ and $q_2$ in $PRL_i$ where $q_1.pri \geq p.pri > q_2$. $E_i$ creates *pseudo-message* $p^*$ for $p$ and appends $p^*$ to the tail of $RRL_i$.

For example, $PRL_i = < b_{[4]} a_{[3]} d_{[3]} c_{[1]}]$ and $RRL_i = < a^* b^* c^* d^*]$ are obtained by inserting $d_{[3]}$ into $PRL_i = < b_{[4]} a_{[3]} c_{[1]}]$ and $RRL_i = < a^* b^* c^*]$. The sequences of *real* messages and pseudo-messages denote the priority-based order and receipt orders of the messages, respectively.

(2) *Pre-acknowledgment*: If $p^* =$ *top* $(RRL_i)$ satisfies PC, $p^*$ is moved from $RRL_i$ to the tail of $PRL_i$.

(3) *Acknowledgment*: If $p=$ *top* $(PRL_i)$ satisfies AC, $p$ is moved to $ARL_i$ and $p^*$ is deleted.

When a lost message $g$ is detected in the PriTO protocol, all the messages following $g$ are removed from $RL_i$ and are retransmitted by the go-back-n and RS. The selective retransmission is adopted in the PriO protocol because the same-priority messages do not need to be totally ordered.

### 5.7 CO Protocol

The CO protocol[20)] provides the CO service by using the MC service. The CO protocol uses the same procedure as the TO except that the selective retransmission is adopted and pre-acknowledged messages are causally ordered. If message $p$ is pre-acknowledged in $E_i$, $p$ is inserted into $PRL_i$ so that the receipt log is causally preserved.

In the CO protocol, $p < q$ iff $q.src \in p.dst$ and the following CO rule hold. Here, $p.src = E_j$.

CO1. $p.tsq < q.tsq$ if $p.src = q.src$.

CO2. $p.tsq < q.ack_j$ if $p.src \neq q.src$.

The CO rule is simpler than ISIS. The CO protocol can not only order the messages in $<$ but also find the lost messages since the CO rule uses the sequence number.

## 6. Evaluation

The group communication protocols are characterized as follows (**Table 2**):

- *System service*.
- *Network service*: 1C, MC, or reliable one.
- *Control scheme*: *centralized*, *decentralized*, and *distributed* ones.
- *Destination*: *selective* and *non-selective*.
- *Communication mode*: *synchronous* and *asynchronous* ones. In the synchronous mode, messages are not sent until the message sent before is atomically received.
- *Retransmission*: *go-back-n* and *selective* schemes.
- *Performance*: The performance is measured in terms of the number of messages transmitted, and the delay time of messages among application processes for

**Table 2** Group communication protocols.

| protocol | system service | network service | cntl. | dst. | mode | recov. | performance | | notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | PDU | delay | |
| GS | TO | 1-to-1/MC | decnt. | group | async. | select. | $n+\epsilon$ | $(h+1)T$ | tree |
| BSS | TO/CO | 1-to-1/OP | decnt. | group | sync. | — | $3n$ | $3T$ | 2 phase |
| KTHB | TO | broadcast/1 C | cnt. | group | sync. | select. | $n+2$ | $2T$ | 2 phase |
| AMp | TO/CO | broadcast/MC | decnt. | group | a./s. | ? | $n+2$ | $3T$ | 2 phase |
| LO | OP | broadcast/MC | dist. | group | async. | select. | $2n+1$ | $3T$ | 3 phase |
| TO | TO/CO | broadcast/1 C | dist. | group | async. | go-back | $2n+1$ | $3T$ | 3 phase |
| CO | CO | broadcast/MC | dist. | group | async. | select. | $2n+1$ | $3T$ | 3 phase |
| SLO | SLO | broadcast/MC | dist. | select. | async. | select. | $2m+1$ | $3T$ | 3 phase |
| STO | STO | broadcast/1 C | dist. | select. | async. | select. | $2m+1$ | $3T$ | 3 phase |
| PriO | PriO | broadcast/1 C | dist. | group | async. | select. | $2n+1$ | $3T$ | 3 phase |
| PriTO | PriTO | broadcast/1 C | dist. | group | async. | go-back | $2n+1$ | $3T$ | 3 phase |

number $n$ of processes (number $m$ ($\leq n$) of destinations in the selective protocols) in the group and the propagation delay time $T$ among processes.

Amoeba[11] uses a centralized protocol (referred to as KTHB) which provides the TO service by using the 1C network. Here, the source process sends message $p$ to the master process *sequencer* and the sequencer broadcasts $p$ to the receivers. When a message loss is detected, the selective retransmission is used. The sequencer supports synchronous communication.

Garcia-Molina[9] presents a decentralized protocol (referred to as GS) based upon the tree structured routing. Each node shows a process and each path denotes a route of message to the destinations. Here, $h$ and $\varepsilon$ denote the height of the tree and the number of processes which are not the destinations in the path, respectively. If an process receives $p$ from the source or the parent, $p$ is relayed to its children until all the destinations receive $p$. The parent decides on the atomic delivery among the children. The lost messages are selectively retransmitted.

ISIS[4] supports CBCAST and ABCAST protocols (referred to as BSS) which provides the CO and TO services, respectively. The vector clock[12] is used to causally order the messages. ABCAST uses a decentralized procedure like two-phase commitment. The LO service is used as the underlying service.

Delta-4[21] supports the atomic multicast protocol (AMp). AMp provides multiple qualities of services (QOS) including the CO and TO services. It adopts the decentralized control. AMp and ISIS discuss how to tolerate stop-failure of the processes.

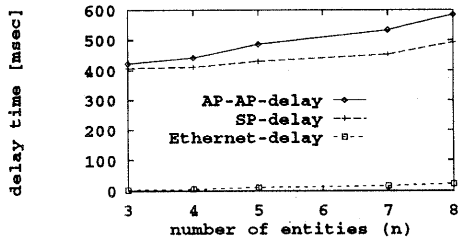


**Fig. 7** Delay time v.s. the number of processes.

We have implemented the protocols as processes of SunOS 4.1* in Sparc2 workstations interconnected by Ethernet. Each workstation has one system process. The process is coded in 5K C steps with object 50K bytes. The delay time is measured in heavy-loaded case, i.e. the application processes send messages continuously to all the processes in the cluster. **Figure 7** illustrates the average delay time of messages for the number $n$ of system processes. *AP-AP-delay* shows time from data request submission of an application process until the receipt of all the destinations. *SP-delay* means how long it takes each message $p$ to be acknowledged after $p$ is accepted. *Ethernet-delay* shows how long it takes to transmit $p$ by using the Ethernet MAC service. Figure 7 shows that delay time is $O(n)$. The delay time is reduced to $O(n)$ by using the piggyback of the acceptance confirmation.

## 7. Concluding Remarks

In this paper, we have presented a model of the group communication service from the data transmission point of view assuming no process failure. We have defined various kinds of group

---

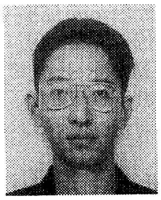
* SunOS is a trademark of Sun Microsystems, Inc.

communication services based on this model. We have also shown group communication protocols which provide the atomic and well-defined ordered delivery of the messages.

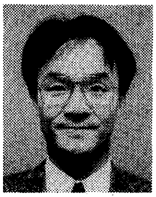
## References

1) Abeysundara, B.W. and Kamal, A. E.: High-Speed Local Area Networks and Their Performance: A Survey, *ACM Comput. Surv.*, Vol.23, No.2, pp. 221-264 (1991).
2) Amir, Y., Dolev, D., Kramer, S. and Malki, D.: Transis: A Communication Sub-System for High Availability, *Proc. of IEEE FTCS-22*, pp. 76-84 (1993).
3) Birman, K. P. and Joseph, T. A.: Reliable Communication in the Presence of Failures, *ACM Trans. Comput. Syst.*, Vol.5, No.1, pp.47-76 (1987).
4) Birman, K., Schiper, A. and Stephenson, P.: Lightweight Causal and Atomic Group Multicast, *ACM Trans. Comput. Syst.*, Vol.9, No.3, pp. 272-314 (1991).
5) Chang, J. M. and Maxemchuk, N.F.: Reliable Broadcast Protocols, *ACM Trans. Comput. Syst.*, Vol.2, No.3, pp. 251-273 (1984).
6) Ellis, C. A., Gibbs, S. J. and Rein, G. L.: Groupware, *Comm. ACM*, Vol.34, No.1, pp.38-58 (1991).
7) Garcia-Molina, H. and Spauster, A.: Message Ordering in a Multicast Environment, *Proc. of IEEE ICDCS-9*, pp. 354-361 (1989).
8) Garcia-Molina, H. and Spauster, A.: Ordered and Reliable Multicast Communication, *ACM Trans. Comput. Syst.*, Vol.9, No.3, pp. 242-271 (1991).
9) Gray, J.: Notes on Database Operating Systems, *Operating Systems: An Advanced Course*, Bayer, R. ed., Springer-Verlag (1978).
10) ISO: OSI—Basic Reference Model, ISO 7493 (1984).
11) Kaashoek, M. F., Tanenbaum, A. S., Hummel, S. F. and Bal, H. E.: An Efficient Reliable Broadcast Protocol, *ACM Operating Systems Review*, Vol.23, No.4, pp. 5-19 (1989).
12) Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System, *Comm. ACM*, Vol.21, No.7, pp.558-565 (1978).
13) Luan, S. W. and Gligor, V. D.: A Fault-Tolerant Protocol for Atomic Broadcast, *IEEE Trans. Parallel and Distributed Systems*, Vol.1, No.3, pp. 271-285 (1990).
14) Mattern, F.: Virtual Time and Global States of Distributed Systems, *Parallel and Distributed Algorithms*, Cosnard, M. and Quinton, P. eds., North-Hollalnd, pp. 215-226 (1989).
15) Melliar-Smith, P. M., Moser, L. E. and Agrawala, V.: Broadcast Protocols for Distributed Systems, *IEEE Trans. on Parallel and Distributed Systems*, Vol.1, No.1, pp. 17-25 (1990).
16) Nakamura, A. and Takizawa, M.: Reliable Broadcast Protocol for Selectively Ordering PDUs, *Proc. of IEEE ICDCS-11*, pp. 239-246 (1991).
17) Nakamura, A. and Takizawa, M.: Design of Reliable Broadcast Communication Protocol for Selectively Partially Ordered PDUs, *Proc. of IEEE COMPSAC'91*, pp. 673-679 (1991).
18) Nakamura, A. and Takizawa, M.: Priority-Based Total and Semi-Total Ordering Broadcast Protocols, *Proc. of IEEE ICDCS-12*, pp. 178-185 (1992).
19) Nakamura, A. and Takizawa, M.: Starvation-Free Priority-Based Total Ordering Broadcast Protocol on High-Speed Single Channel Network, *Proc. of IEEE HPDC-2*, pp. 281-288 (1993).
20) Nakamura, A. and Takizawa, M.: Causally Ordering Broadcast Protocol, *Proc. of IEEE ICDCS-14*, pp.48-55 (1994).
21) Powell, D., Chereque, M. and Drackley, D.: Fault-Tolerance in Delta-4, *ACM Operating Systems Review*, Vol.25, No.2, pp. 122-125 (1991).
22) Ravindran, K. and Shah, K.: Causal Broadcasting and Consistency of Distributed Shared Data, *Proc. of IEEE ICDCS-14*, pp. 40-47 (1994).
23) Schneider, F. B., Gries, D. and Schlichting, R. D.: Fault-Tolerant Broadcasts, *Science of Computer Programming*, Vol.4, No.1, pp. 1-15 (1984).
24) Tachikawa, T. and Takizawa, M.: Selective Total Ordering Broadcast Protocol, *Proc. of IEEE ICNP'94*, pp. 212-219 (1994).
25) Takizawa, M.: Cluster Control Protocol for Highly Reliable Broadcast Communication, *Proc. of IFIP Conf. on Distributed Processing*, pp. 431-445 (1987).
26) Takizawa, M. and Nakamura, A.: Partially Ordering Broadcast (PO) Protocol, *Proc. of IEEE INFOCOM'90*, pp. 357-364 (1990).
27) Takizawa, M. and Nakamura, A.: Reliable Broadcast Communication, *Proc. of InfoJapan*, pp. 325-332 (1990).
28) Verissimo, P., Rodrigues, L. and Baptista, M.: AMp: A Highly Parallel Atomic Multicast Protocol, *Proc. of ACM SIGCOMM'89*, pp. 83-93 (1989).

(Received September 1, 1994)
(Accepted May 12, 1995)



**Takayuki Tachikawa** received his B. E. degree in computers and systems engineering from Tokyo Denki University, Japan in 1994. Currently, he is studying towards the M. E. degree at computers and systems engineering, Tokyo Denki University. His research interests include distributed systems, computer networks, and communication protocols. He is a member of IPSJ.



**Makoto Takizawa** received his B. E. and M. E. degrees in Applied Physics from Tohoku University, Japan, in 1973 and 1975, respectively. He received his D. E. in Computer Science from Tohoku University in 1983. From 1975 to 1986, he worked for Japan Information Processing Developing Center (JIPDEC) supported by the MITI. He was an associate professor since 1987, and he is currently a Professor of the Department of Computers and Systems Engineering, Tokyo Denki University since 1994. From 1989 to 1990, he was a visiting professor of the GMD-IPSI, Germany. He is also a regular visiting professor of Keele University, England since 1990. His research interest includes communication protocols, group communication, distributed database systems, transaction management, and groupware. Prof. Takizawa is a member of IEEE, ACM, IPSJ, and IEICE.